**Wireshark and Power Bi Network Traffic Analysis Project**

**Summary**: To compare a normal network packet capture with a compromised network packet capture and analyze the differences. Highlight differences by creating network visualizations in Power BI.

**Step 1: Packet Capture**

First, I used Wireshark to perform a packet capture on my network. I used this file as a baseline for normal network traffic. I extracted columns from the packet capture file into a CSV file using tshark. The columns included by Wireshark in the PCAP to CSV export were not sufficient for my analysis, so I used tshark to extract more data from the packet capture. To export my network’s packet capture to a CSV file, I used the command:

tshark -r "C:\Users\jenna\Documents\fourth.pcapng" -T fields -E separator=, -E header=y -e frame.time\_epoch -e ip.src -e ip.dst -e ip.proto -e tcp.srcport -e tcp.dstport -e udp.dstport -e frame.len -e tcp.flags.syn -e tcp.flags.ack -e dns.qry.name -e dns.a -e ipv6.src -e ipv6.dst > good\_pcap.csv

I extracted the columns for time, source/destination IP addresses and ports, and TCP flags. This information did not end up being enough to analyze the traffic, so in Power BI I had to import and merge the columns for the packet’s bytes and protocols into the tables I made from the CSV files.

Next, I found a public PCAP sample file that captured traffic from a device that was infected with the Emotet malware from malware-traffic-analysis.net. The victim computer became infected from downloading a Macro Word document, which subsequently caused a lot of HTTPS traffic between itself and a C2 server.

To export the emotet’s packet capture information to a CSV file, I used the command:

tshark -r "C:\Users\jenna\Downloads\emotet.pcap" -T fields -E separator=, -E header=y -e frame.time\_epoch -e ip.src -e ip.dst -e ip.proto -e tcp.srcport -e tcp.dstport -e udp.dstport -e frame.len -e tcp.flags.syn -e tcp.flags.ack -e dns.qry.name -e dns.a -e ipv6.src -e ipv6.dst -e http.content\_type -e http.request.method -e http.user\_agent -e http.request.uri -e tls.handshake.type -e tls.handshake.extensions\_server\_name -e tls.handshake.session\_id -e http.host -e tcp.time\_delta -e tcp.analysis.retransmission > bad\_pcap.csv

I extracted more columns from the infected PCAP file to try to learn more about the HTTP traffic.

**Step 2: PCAP Analysis**

From my initial examination of the Emotet packet capture, there was a substantially larger amount of TCP traffic in the Emotet PCAP than in my network’s PCAP. According to my analysis in Power BI, the Emotet’s packet capture was made up of 94% TCP traffic, while my network only contained 15% TCP traffic.

I found the packet stream where the infected Word file was downloaded by filtering with “http.request.method”. I selected the first “GET” request and followed the HTTP stream. It showed that the Word document was downloaded.

A screenshot of a computer

AI-generated content may be incorrect.

After this file was downloaded, there was a large amount of encrypted traffic between the host computer, 10.1.21.101 and a server named sale.mandinipearls.com on non-standard TCP ports, such as 4970. There were many TCP Keep Alive messages and ACKs between the infected device and this website. This server was likely the C2(Command and Control) Server, and the traffic represented updates and instructions for the infected host.

The packet capture for my network contained a variety of protocols including TLS, MDNS, ARP and QUIC (transport layer protocol). There was a lot of standard network traffic, such as devices communicating with the router using ARP to find out different hosts’ MAC addresses or using DNS to resolve IP addresses. There was some application data exchanged over the network using TLS, but since it was encrypted and I was unable to decrypt it using my sslhostkey file, I was unable to read it.

**Step 3: Import Data and Analyze with Power BI**

I imported 2 CSV files into Power BI, one for each PCAP file that I extracted information from using tshark. First, I cleaned up and formatted the data. There were several blanks in columns, such as the IP source and destination address, and I had to decide which blanks to fill with 0s or leave empty. I decided to fill in the IP addresses with 0s and leave other cells blank, such as the TCP SYN/ACK flags, because they were not necessary for any calculations or for my charts/graphs. I also formatted the IP source and destination columns as text instead of as a decimal number.

I created a column to calculate the date and time for each packet, from the time\_epoch column information from Wireshark, using the formula: (time\_epoch/86400)+DATE(1970,1,1).

I created a column to classify the direction of the traffic, either as incoming or outgoing with the formulas:

**For the Normal PCAP:**

Table.AddColumn(#"Changed Type", "Direction", each let

src=Text.From([ip.src]),

dst=Text.From([ip.dst])

in

if src ="0" or dst ="0" then "Unknown"

else if Text.StartsWith(src,"192") then "Outgoing"

else "Incoming")

**For the Emotet/Botnet PCAP:**

Table.AddColumn(#"Changed Type", "Direction", each let

src=Text.From([ip.src]),

dst=Text.From([ip.dst])

in

if src ="0" or dst ="0" then "Unknown"

else if Text.StartsWith(src,"10") then "Outgoing"

else "Incoming")

If the traffic started with an octet in the private IP range (10 or 192), then I classified those packets as being “Outgoing” traffic. If either the source or destination IP was 0, it was “Unknown” traffic, and everything else was classified as “Incoming” traffic.

I added columns that I imported from a CSV to include the packet bytes and protocol distribution from Wireshark into Power BI using Transform Data. I added index columns on the PCAP tables and the tables with the columns I wanted to import and merged the columns into the existing PCAP tables on the index column.

I calculated the number of packets by using the countrow() function on both tables. I also calculated the sum and average size of the packet bytes using the sum() and average() functions.

Because there was such a big difference in the number of packets captured between my network and the Emotet/botnet PCAP, I calculated the percentage of each type of protocol out of all the traffic instead of counting the number of packets of each protocol type. To create a column for the percent of packets by protocol, I used the formula

DIVIDE(

SUM('bad\_pcap'[bytes\_bad.Packet Bytes]),

CALCULATE(

SUM('bad\_pcap'[bytes\_bad.Packet Bytes]),

ALL('bad\_pcap')

)

)

I used the percent of packets column for the X axis and the protocols for Y axis to create a bar chart that shows the percent of packets by each type of protocol.

The next chart I made was a line chart that showed the number of packets captured for each 10 second interval for the normal network and for each 1-minute interval for the Emotet/botnet network. I chose to use a 10 second interval for my network’s PCAP because I captured packets for a much shorter amount of time.

To count the number of packets for each 10 second- and 1-minute interval, I used the formulas:

Packets\_Per\_Minute = (BAD PCAP)

COUNTROWS(

FILTER(

'bad\_pcap',

'bad\_pcap'[Time\_Rounded\_Minute] = MAX('bad\_pcap'[Time\_Rounded\_Minute])

)

)

Packets\_Per\_10s = (GOOD PCAP)

COUNTROWS(

FILTER(

'good\_pcap',

'good\_pcap'[Time\_Rounded\_10\_good] = MAX('good\_pcap'[Time\_Rounded\_10\_good])

)

)

I also made a stacked bar chart that shows the number of bytes and packets for the top 3 IP address conversations.

Last, I made a table that provided a summary of the total number of bytes, packets and frame lengths by protocol for each network.